MULTI-MODAL WEB INTERACTION
OVER WIRELESS NETWORK

FIELD OF INVENTION

5   This invention relates to web interaction over a wireless network between wireless communication devices and an Internet application. Particularly, the present invention relates to multi-modal web interaction over wireless network, which enables users to interact with an Internet application in a variety of ways.

BACKGROUND OF THE INVENTION

10   Wireless communication devices are becoming increasingly prevalent for personal communication needs. These devices include, for example, cellular telephones, alphanumeric pagers, "palmtop" computers, personal information managers (PIMS), and other small, primarily handheld communication and computing devices. Wireless

15   communication devices have matured considerably in their features and now support not only basic point-to-point communication functions like telephone calling, but more advanced communications functions, such as electronic mail, facsimile receipt and transmission, Internet access and browsing of the World Wide Web, and the like.

Generally, conventional wireless communication devices have software that manages

20   various handset functions and the telecommunications connection to the base station. The software that manages all the telephony functions is typically referred to as the telephone stack. The software that manages the output and input, such as key presses and screen display, is referred to as the user interface or Man-Machine-Interface or "MMI.

25   U.S. Patent No. 6,317,781 discloses a markup language based man-machine interface. The man-machine interface provides a user interface for the various telecommunications functionality of the wireless communication device, including dialing telephone numbers, answering telephone calls, creating messages, sending messages, receiving messages, and establishing configuration settings, which are defined in a well-known markup language,

30   such as HTML, and accessed through a browser program executed by the wireless communication device. This feature enables direct access to Internet and World Wide web content, such as web pages, to be directly integrated with telecommunication functions of the device, and allows web content to be seamlessly integrated with other types of data, because

1

all data presented to the user via the user interface is presented via markup language-based pages. Such a markup language based man-machine interface enables users directly to interact with an Internet application.

5　　　However, unlike conventional desktop or notebook computers, wireless communication devices have a very limited input capability. Desktop or notebook computers have cursor based pointing devices, such as computer mouse, trackballs, joysticks, and the like, and full keyboards. This enables navigation of Web content by clicking and dragging of scroll bars, clicking of hypertext links, and keyboard tabbing between fields of forms, such as HTML forms. In contrast, wireless communication devices have a very

10　　limited input capability, typically up and down keys, and one to three soft keys. Thus, even with a markup language based man-machine interface, users of wireless communication devices are unable to interact with an Internet application using conventional technology. Although some forms of speech recognition exist in the prior art, there is no prior art system to realize multi-modal web interaction, which will enable users to perform web interaction

15　　over a wireless network in a variety of ways.

BRIEF DESCRIPTION OF THE DRAWINGS

The features of the present invention will be more fully understood by reference to the accompanying drawings, in which:

20　　　Figure 1 is an illustration of the network environment in which an embodiment of the present invention may be applied.

Figure 2 is an illustration of the system 100 for web interaction over a wireless network according to one embodiment of the present invention.

Figure 3 and Figure 4 show focus on a group of hyperlinks or a form.

25　　　Figures 5-6 present the MML event mechanisms.

Figure 7 presents the fundamental flow chart of system messages & MML events.

Figure 8 shows the details of MML element blocks used in the system of one embodiment of the present invention.

30　DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be

appreciated by one of ordinary skill in the art that the present invention shall not be limited to these specific details.

Various embodiments of the present invention overcome the limitation of the conventional Man-Machine Interface for wireless communication by providing a system and
5   method for multi-modal web interaction over a wireless network. The multi-modal web interaction of the present invention will enable users to interact with an Internet application in a variety of ways, including, for example:

Input: Keyboard, Keypad, Mouse, Stylus, speech;

Output: Plaintext, Graphics, Motion video, Audio, Synthesis speech.

10 Each of these modes can be used independently or concurrently. In one embodiment described in more detail below, the invention uses a multi-modal markup language (MML).

In one embodiment, the present invention provides an approach for web interaction over wireless network. In the embodiment, a client system receives user inputs, interprets the user inputs to determine at least one of several web interaction modes, produces a
15 corresponding client request and transmits the client request. The server receives and interprets the client request to perform specific retrieving jobs, and transmits the result to the client system.

In one embodiment, the invention is implemented using a multi-modal markup language (MML) with DSR (Distributed Speech Recognition) mechanism, focus mechanism,
20 synchronization mechanism and control mechanism, wherein the focus mechanism is used for determining which active display is to be focused and the ID of the focused display element. The synchronization mechanism is used for retrieving the synchronization relation between a speech element and a display element to build the grammar of corresponding speech element to deal with user's speech input. The control mechanism controls the
25 interaction between client and server. According to such an implementation, the multi-modal web interaction flow is shown by way of example as follows:

User: point and click using hyperlinks, submit a form (traditional web interaction) or press the "Talk Button" and input an utterance (speech interaction).

Client: receive and interpret the user input. In case of traditional web interaction, the client
30 transmits a request to the server for a new page or submits the form. In case of speech interaction, the client determines which active display element is to be focused and the identifier (ID) of the focused display element, captures speech, extracts speech features, and transmits the speech features, the ID of focused display element and other information such

as URL of the current page to the server. The client waits for the corresponding server response.

Server: receive and interpret the request from the client. In the case of traditional web interaction, the server retrieves the new page from cache or web server and sends the page to the client. In the case of speech interaction, the server receives the ID of the focused display element to build the correct grammar based on the synchronization of display elements and speech elements. Then, speech recognition will be performed. According to the result of speech recognition, the server will do specific jobs and send events or new page to the client. Then, the server waits for new requests from the client.

Client: load the new page or handle events.

The various embodiments of the present invention described herein provide an approach to use Distributed Speech Recognition (DSR) technology to realize multi-modal web interaction. The approach enables each of several interaction modes to be used independently or concurrently.

As a further benefit of the present invention, with the focus mechanism and synchronization mechanism, the present invention will enable the speech recognition technology to be feasibly used to retrieve information on the web, improve the precision of speech recognition, reduce the computing resources necessary for speech recognition, and realize real-time speech recognition.

As a further benefit of the present invention, with one implementation based on a multi-modal markup language, which is an extension of XML by adding speech features, the approach of the present invention can be shared across communities. The approach can be used to help Internet Service Providers (ISP) to easily build server platforms for multi-modal web interaction. The approach can be used to help Internet Content Providers (ICP) to easily create applications with the feature of multi-modal web interaction. Specifically, Multi-modal Markup Language (MML) can be used to develop speech applications on the web for at least two scenarios:

Multi-modal applications can be authored by adding MML to a visual XML page for a speech model;

Using the MML features for DTMF input, voice-only applications can be written for scenarios in which a visual display is unavailable, such as the telephone.

This allows content developers to re-use code for processing user input. The application logic remains the same across scenarios: the underlying application does not need to know whether the information is obtained by speech or other input methods.
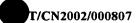
Referring now to Figure 1, there is shown an illustration of the network environment

5   in which an embodiment of the invention may be applied. As shown in Figure 1, client 10 can access documents from Web server 12 via the Internet 5, particularly via the World-Wide Web ("the Web"). As well known, the Web is a collection of formatted hypertext pages located on numerous computers around the world that are logically connected by the Internet. The client 10 may be personal computers or various mobile computing devices 14,

10   such as personal digital assistants or wireless telephones. Personal digital assistants, or PDA's, are commonly known hand-held computers that can be used to store various personal information including, but not limited to contact information, calendar information, etc. Such information can be downloaded from other computer systems, or can be inputted by way of a stylus and pressure sensitive screen of the PDA. Examples of PDA's are the

15   Palm™ computer of 3Com Corporation, and Microsoft CE™ computers, which are each available from a variety of vendors. A user operating a mobile computing device such as a cordless handset, dual-mode cordless handset, PDA or operating portable laptop computer generates control commands to access the Internet. The control commands may consist of digitally encoded data, DTMF or voice commands. These control commands are often

20   transmitted to a gateway 18. The gateway 18 processes the control commands (including performing speech recognition) from the mobile computing device 14 and transmits requests to the Web server 12. In response to the request, the Web server 12 sends documents to the gateway 18. Then, the gateway 18 consolidates display contents from the document and sends the display contents to the client 14.

25   According to an embodiment of the present invention for web interaction over wireless network, the client 14 interprets the user inputs to determine a web interaction mode, produces and transmits the client 14 request based on the interaction mode determination result; and multi-modal markup language (MML) server (gateway) 18 interprets the client 14 request to perform specific retrieving jobs. The Web interaction mode can be traditional

30   input/output (for example: keyboard, keypad, mouse and stylus/plaintext, graphics, and motion video) or speech input/audio (synthesis speech) output. This embodiment enables users to browse the World Wide Web in a variety of ways. Specifically, users can interact with an Internet application via traditional input/output and speech input/output

5

independently or concurrently.

In the following section, we will describe a system for web interaction over a wireless network according to one embodiment of the present invention. The reference design we give is one implementation of MML. It extends XHTML Basic by adding speech features to enhance XHTML modules. The motivation for XHTML Basic is to provide an XHTML document type that can be shared across communities. Thus, an XHTML Basic document can be presented on the maximum number of Web clients, such as mobile phones, PDAs and smart phones. That is the reason to implement MML based on XHTML Basic.

XHTML Basic Modules in one embodiment:

Structure Module; Text Module; Hypertext Module; List Module; Basic Forms Module; Basic Tables Module; Image Module; Object Module; Metainformation Module; Link Module and Base Module.

Other XHTML Modules can provide more features:

Script Module: Support client side script.

Style Module: Support inline style sheet.

Referring to Figure 2, there is shown an illustration of the system 100 for web interaction over a wireless network according to one embodiment of the invention. In Figure 2, only the components related to the present invention are shown so as not to obscure the invention. As shown in Figure 2, the client 110 comprises: web interaction mode interpreter 111, speech input/output processor 112, focus mechanism 113, traditional input/output processor 114, data wrap 115 and control mechanism 116. The MML server 120 comprises: web interaction mode interpreter 121, speech recognition processor 122, synchronization mechanism 123, dynamic grammar builder 124, HTTP processor 125, data wrap 126 and control mechanism 127.

In the system 100, at client 110, web interaction mode interpreter 111 receives and interprets user inputs to determine the web interaction mode. The web interaction mode interpreter 111 also assists content interpretation in the client 110. In case of traditional web interaction, traditional input/output processor 114 processes user input, then data wrap 115 transmits a request to the server 120 for a new page or form submittal. In case of speech interaction, speech input/output processor 112 captures and extracts speech features, focus

mechanism 113 determines which active display element is to be focused upon and the ID of the focused display element. Then data wrap 115 transmits the extracted speech features, the ID of the focused display element and other information such as URL of current page to the MML server. At MML server 120: web interaction mode interpreter 121 receives and interprets the request from the client 110 to determine the web interaction mode. The web interpretation mode interpreter 121 also assists content interpretation on the server 120. In case of traditional web interaction, HTTP processor 125 retrieves the new page or form from cache or web server 130. In case of speech interaction, synchronization mechanism 123 retrieves the synchronization relation between a speech element and a display element based on the received ID, dynamic grammar builder 124 builds the correct grammar based on the synchronization relation between speech element and display element. Speech recognition processor 122 performs speech recognition based on the correct grammar built by dynamic grammar builder 124. According the recognition result, HTTP processor 125 retrieves the new page from cache or web server 130. Then, data wrap 126 transmits a response to the client 110 based on the retrieved result. The control mechanisms 116 and 127 are used to control the interaction between the client and the server.

The following section is a detailed description of one embodiment of the present invention using MML with a focus mechanism, synchronization mechanism and control mechanism according to the embodiment.

Focus Mechanism

In multi-modal web interaction, besides traditional input methods, speech input can become a new input source. When using speech interaction, speech is detected and feature is extracted at the client, and speech recognition is performed at the server. We note that generally, the user will typically do input using the following types of conventional display element(s):

Hyperlinks: The user can select the hyperlink(s) that is stable.

Form: The user can view and/or modify an electronic form containing information such as stock price, money exchange, flights and the like.

Considering the limitations of current speech recognition technology, in the multi-modal web interaction of the present invention, a focus mechanism is provided to focus the user's attention on the active display element(s) on which the user will perform speech input. A display element is focused by highlighting or otherwise rendering distinctive the display

element upon which the user's speech input will be applied. When the identifier (ID) of the focused display element(s) is transmitted to the server, the server can perform speech recognition based on the corresponding relationship between the display element and the speech element. Therefore, instead of conventional dictation with a very large vocabulary,

5   the vocabulary database of one embodiment is based on the hyperlinks, electronic forms, and other display elements on which users will perform speech input. At the same time, at the server, the correct grammar can be built dynamically based on the synchronization of display elements and speech elements. Therefore, the precision of speech recognition will be improved, the computing load of speech recognition will be reduced, and real-time speech

10   recognition will actually be realized.

       The MMI's of Figure 3 and Figure 4 can help to understand the focus processing on the active display element of one embodiment. Figure 3 shows the focus on a group of hyperlinks, and Figure 4 shows the focus on a form.

       In the conventional XHTML specification, it is not allowed to add a BUTTON out of

15   a form. As our strategy is not to change the XHTML specification, the "Programmable Hardware Button" is adopted to focus a group of hyperlinks in one embodiment. The software button with title of "Talk to Me" is adopted to focus the electronic form display element. It will be apparent to one of ordinary skill in the art that other input means may equivalently be associated with focus for a particular display element.

20   When a "card" or page of a document is displayed on the display screen, no display element is initially focused. With the "Programmable Hardware Button" or "Talk to Me Button", the user can perform web interaction through speech methods. If the user activates the "Programmable Hardware Button" or "Talk To Me Button ", the display element(s) to which the button belongs is focused. Then, possible circumstances might be as follows:

25

User speech

       Once a user causes focus on a particular display element, an utterance from the user is received and scored or matched against available input selections associated with the focused display element. If the scored utterance is close enough to a particular input

30   selection, the "match" event is produced and new card or page is displayed.

       The new card or page corresponds to the matched input selection. If the scored utterance cannot be matched to a particular input selection, a "no match" event is produced, audio or text prompt is displayed and the display element is still focused.

The user may also use traditional ways of causing a particular display element to be focused, such as pointing at an input area, such as a box in a form. In this case, the currently focused display element changes into un-focused as a different display element is selected.

The user may also point to a hypertext link, which causes a new card or page to be

5 displayed. If the user points the other "Talk To Me Button", the previous display element changes into un-focused and the display element, to which the last activation belongs, changes into focused.

If the user does not do anything for longer than the length of a pre-configured timeout, the focused display element may change into un-focused.

10

Synchronize Mechanism

When the user wishes to provide input on a display element through a speech methodology, the grammar of the corresponding speech elements should be loaded at the server to deal with the user's speech input. So, the synchronization or configuration scheme for the speech

15 element and the display element is necessary. Following are two embodiments which accomplish this result.

One fundamental speech element has one grammar that includes all entrance words for one time speech interaction on the Web.

20 One of the fundamental speech elements must have one and only one corresponding display element*(s)* as follows:

One group of hyperlinks

One Form

One identified single or group of display elements.

25 Thus, it is necessary to perform a binding function to bind speech elements to corresponding display elements. In one embodiment, a "bind" attribute is defined in *<mml:link>*, *<mml:sform>* and *<mml:input>*. It contains the information for one pair of display elements and corresponding speech element.

The following section presents sample source code for such a binding for hyperlink

30 type display elements.

```
<mml:card>
        <a id="stock">stock</a>
```

```
        <a id="flight">flight</a>
        <a id="weather">weather</a>
         <mml:speech>
            <mml:recog>
                <mml:group>
                     <mml grammar src="grammar.gram" />
                     <mml:link value="#stock-gram" bind="stock"/>
                     <mml:link value="#flight-gram"
bind="flight"/>
                     <mml:link value="#weather-gram"
bind="weather"/>
                </mml:group>
            </mml:recog>
         </mml:speech>
      </mml:card>
```

The following section presents sample source code for a binding in an electronic form, such as an airline flight information form, for example.

```
<mml:card title="flight inquery">
    <script language="javascript">
       function talk1()
       {
          var sr= new DSR.FrontEnd;
          sr.start("form-flight");
       }
    </script>


    <p>flightquery: </p>

    <form id="form-flight" action="flightquery.asp"
method="post">
        <p>date: <input type="text" id ="date1" name="date01"/>
```

```
        company(optional):<input type="text" id ="company1"
name="company01"/>
        </p>
        <p>startfrom: <input type="text" id ="start-from"
name="start"/>
        arrivingat:<input type="text" id ="arriving-at"
name="end"/>
        </p>
        <p><input   type="submit"  value="submit"/>  <input   type="Reset"
value="Reset"/>
        <input type="button" value="Talk To Me"
onclick="talk1()"/>
        </p>
</form>

<mml:speech>
    <mml:recog>

        <mml:sform id="sform-flight" bind="form-flight">
            <mml:grammar src="flight-query.gram"/>
            <mml:input id="sdate" value="#date" bind="date1"/>
            <mml:input id="scompany" value="#company"
bind="company1"/>
            <mml:input id="sstart" value="#start" bind="start-
from"/>
            <mml:input id="send" value="#end" bind="arriving-
at"/>
            <mml:onevent type="match">
                <mml:do target="flight-prompt" type="activation"/>
            </mml:onevent>
        </mml:sform>
        <mml:onevent type="nomatch">
            <mml:do target="prompt1" type="activation"/>
        </mml:onevent>
```

```
        </mml:recog>
    </mml:speech>
</mml:card>
```

5   **Client-Server Control Mechanism**

When performing multi-modal interaction, in order to signal the user agent and server that some actions have taken place, the system messages produced at the client or the server or other events produced at the client or the server should be well defined.

In an embodiment of the present invention, a Client-Server Control Mechanism is 10   designed to provide a mechanism for the definition of the system messages and MML events which are needed to control the interaction between the client and server.

Table 1 includes a representative set of system messages and MML events.

	System Messages	Events	Communicated between client and server
Error (Server)	√		√
Transmission (Server)	√		√
Transmission (Client)	√		√
Ready (Server)	√		√
Session (Client)	√		√
Exit (Client)	√		√
OnFocus*(Client)	√		√
UnFocus*(Client)	√		
Match (Server)		√	√
Nomatch (Server)		√	√
Onload (Client)		√	
Unload (Client)		√	

Table 1 Control Information Table

15

System Messages:

The System Messages are for client and server to exchange system information. Some types of system Messages are triggered by the client and sent to the server. Others are triggered by the server and sent to the client.

5   In one embodiment, the System messages triggered at the client include the following:

<1>   Session message

The Session message is sent when the client initializes the connection to the server. A Ready message or an Error Message is expected to be received from the server after the Session Message is sent. Below is the example of the Session Message:

10

```
<message type="session">
        <ip>    </ip >              <!-- the IP address of the requesting client -->
        <type>  </type >            <!-- the device type of the client -->
        <voice> </voice>            <!-- the voice characters of the user -->
15                          <!-- such as man, woman, old man, old woman, child -- >
        <language> </language>      <!-- the language that the user speak -->
        <accuracy> </accuracy>      <!-- the default recognition accuracy that the
client request -->
</message>
```

20

<2> Transmission message

The Transmission Message (Client) is sent after the client establishes the session with the server. A Transmission Message (Server) or an Error Message is expected to be received from the server after the Transmission Message (Client) is sent. Below is an
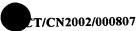
25   example of the Transmission message:

```
<message type="transmission">
        <session> </session>               <!-- the ID of this session-->
        < crc>   </crc>              <!-- the crc information that the client
30   requests-->
        <QoS>   </QoS>               <!-- QoS information that the client requests-->
        <bandwith></bandwith>              <!-- Bandwith that the client requests-->
</message>
```

35

<3>   OnFocus message

OnFocus and UnFocus messages are special client side System Messages.

OnFocus occurs when user points on, or presses, or otherwise activates the "Talk Button" (Here "Talk Button" means "Hardware Programmable Button" and "Talk to Me Button"). When OnFocus occurs, the client will perform the following tasks:

   a.   Open the microphone and do Front-end detection

5 b.   When the start-point of real speech is captured, do front-end speech processing. The ID of the corresponding focused display element and other essential information (e.g. the URL of current page) is transmitted to the server with the first packet of speech features.

   c.   When the first packet of speech features reach the server, the corresponding
10 grammar will be loaded into the recognizer and speech recognition will be performed.

Below is an example of the OnFocus message to be transmitted to the server:

```
    <message type="OnFocus">
15      <session> <session>    <!--the ID of this session -->
        <id>   </id>                <!--the ID of the focused display element -->
        <url>  <url>    .           <!--the URL of the document that is loaded by
    client now -->
    </message>
20
```

It is recommended that the OnFocus Message be transmitted with speech features rather than transmitted alone. The reason is to optimize and reduce unnecessary communication and server load in these cases:

   "When the user switches and presses two different "Talk Buttons" in one card or on
25 one page before entering one utterance, the client will send an unnecessary OnFocus Message to the server and will cause the server to build a grammar unnecessarily."

   But a software vendor can choose to implement the OnFocus Message as transmitted alone.

30 <4>   UnFocus Message

   When UnFocus occurs, the client will perform the task of closing the microphone. UnFocus occurs in the following cases:

   a.   User points on, presses or otherwise activates the "Talk Button" of the focused display element.

b.   User uses a traditional way like a pointer to point at the input areas such as a box of form and the like.

In the below case, UnFocus will not occur,

a.   User points on, presses or otherwise activates the "Talk Button" of the unfocused display element while there is a focused display element in the card or page.

<5>   Exit message

The Exit Message is sent when the client quits the session. Below is an example:

```
<message type="exit">
    <session>    <session>    <!-- the ID of this session -->
</message>
```

System Messages Triggered at the Server

<1>   Ready message

The Ready Message is sent by the server when the client sends the Session Message first and the server is ready to work. Below is an example:

```
<message type="ready">
    <session> </session>       <!-- the ID of this session which is created by server
after -->
                               <!-- the Session Message is received-->
    <ip></ip>                  <!-- the IP address of the responding server -->
    <voice>
        <support>T</support>  <!-- T or F, the server supports or not support the voice
character-->
                               <!-- that the client request in Session Message -->
        <server>  </server>    <!-- the voice character that the server is using now--
>
    </voice>
    <language>
        <support>T</support>  <!-- T or F, the server supports or not support the
language-->
                                   <!-- that the client request in Session
Message -->
        <server> </server>    <!-- the language that the server is using now-->
```
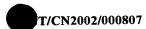
```
     </language>
     <accuracy>
        <support>T</support>   <!-- T or F, the server supports or not support the
default-->
                                  <!--recognition accuracy that the client request in
Session Message -->
        <server> </server>   <!-- the default recognition accuracy that the server is
using now-->
     </accuracy>
   </message>
```

<2>Transmission message

The Transmission message is sent by the server when the client sends a transmission message first or the network status has changed. This message is used to notify the client of the transmission parameters the client should use. Below is an example:

```
<message type="transmission">
        <session>      </session>   <!-- the ID of this session-->
  < crc>            </crc>            <!-- the crc information -->
        <QoS>            </QoS>         <!-- QoS information -->
          <bandwidth> </bandwidth>    <!--the bandwidth which client should use-->
</message>
```

<3>   Error message

The Error message is sent by the server. If the server generates some error while processing the client request, the server will send an Error Message to the client. Below is an example:

```
<message type="error">
    <session>          </session>    <!-- the ID of this session-->
    <errorcode> 500              </errorcode>     <!-- the code number of the error --
>
    <errorinfo>     </errorinfo>         <!-- the text information of the error-->
</message>
```

MML Events

The purpose of MML events is to supply a flexible interface framework for handling various processing events. MML events can be categorized as client-produced events and

server-produced events according to the event source. And the events might need to be communicated between the client and the server.

In the MML definition, the element of event processing instruction is **<mml:onevent>**. There are four types of events:

5 **Events trigged at server**

<1> Match Event

When speech processing results in a match, if the page developer adds the processing instructions in the handler of the "nomatch" event in the MML page,

10
```
<mml:card>
    <form id="form01" action="other.mml" method="post">
        <input id="text1" type="text" name="add" value="上海"/>
    </form>
    ...
15

    <mml:speech>
        <mml:prompt id="promptServer" type="tts">
            The place you want to go <mml:getvalue from="stext1" at="server" />
        </mml:prompt>
20      <mml:recog>
            <mml:sform id="sform01" bind="form01">
                <mml:grammar src="Add.gram"/>
                <mml:input id="stext1" value="#add" bind="text1"/>
                <mml:onevent type="match">
25                  <mml:do target="promptServer" type="activation"/>
                </mml:onevent>
            </mml:sform>
        </mml:recog>
    < /mml:speech>
30  </mml:card>
```

the event is sent to the client as in the following example:

```
<event type="match">
35      <do target="promptServer">
        <input id="stext1" value="place"/>  <!--It's according to the
recognition result -->
</event>
```

40   If the page developer doesn't handle the "match", no event is sent to the client.

<2> Nomatch Event

When speech processing results in a non-match, if the page developer adds the processing instructions in the handler of the "nomatch" event in the MML page,

5

```
<mml:onevent type="nomatch">
      <mml:do target="prompt1" type="activate"/>
</mml:onevent>
```

10

the event is sent to the client as in the following example:

```
<event type="nomatch">
       <do target = "prompt1"/> <!-- Specified by page developer -->
```
15   `</event>`

If the page developer doesn't handle the "nomatch", the event is sent to the client as follows:

20   `<event type="nomatch"/>`

Events trigged at client

25   <1> Onload Event

The "Onload" event occurs when certain display elements are loaded. This event type could only be valid when the **trigger** attribute is sent to the "client". The page developer could add the processing instructions in the handler of the "Onload" event in the MML page:

30

```
<mml:onevent type="onload">
             <mml:do target="prompt1" type="activate"/>
</mml:onevent>
```

35   No "Onload" event needs to be sent to the server.

<2> Unload Event

The "Unload" event occurs when certain display elements are unloaded. This event type could only be valid when the trigger attribute is sent to the "client". The page developer could add the processing instructions in the handler of the "Onload" event in the MML page,

```
<mml:onevent type="unload">
        <mml:do target="prompt1" type="activate"/>
</mml:onevent>
```

No "Onload" event need be sent to the server.

MML Events Conformance

The MML Events Mechanism of one embodiment is an extension of the conventional XML Event Mechanism. As shown in Figure 5, there are two phases in the conventional event handling: "capture" and "bubbling" (See XML Event Conformance).

To simplify the Event mechanism, to improve efficiency, and to ease the implementation more, we developed the MML Simple Events Mechanism of one embodiment. As shown in Figure 6, in the Simple Event Mechanism, neither a "capture" nor "bubbling" phase is needed. In the MML Event Mechanism of one embodiment, the observer node must be the parent of the event handler <mml:onevent>. The event triggered by one node is to be handled only by the child <mml:onevent> event handler node. Other <mml:onevent> nodes will not intercept the event. Further, the phase attribute of <mml:onevent> is ignored.

Figures 5 and 6 illustrate the two event mechanisms. The dotted line parent node (node 510 in Figure 5 and node 610 in Figure 6) will intercept the event. The child node <mml:onevent> (node 520 in Figure 5 and node 620 in Figure 6) of the dotted line node will have the chance to handle the specific events.

MML Events in the embodiment have the unified event interface with the host language (XHTML) but are independent from traditional events of the host language. Page developers can write events in the MML web page by adding a *<mml:onevent>* tag as the child node of an observer node or a target node.

Figure 7 illustrates the fundamental flow chart of the processing of system messages & MML events used in one embodiment of the present invention. This processing can be partitioned into the following segments with the listed steps performed for each segment as

shown in Figure 7.

1) Connection:

Step 1: Session message is sent from client to server

5 Step 2: Ready message is sent from server to client

Step 3: Transmission message (transmission parameters) is sent from client to server

Step 4: Transmission message (transmission parameters) is sent from server to client

When a mismatch occurs in the above four steps, an error message will be sent to the

10 client from the server.

2) Speech Interaction:

Step 1: Feature Flow with OnFocus message is sent from the client to the server

Step 2: Several cases will happen:

15 Result match:

If the implementation includes optional event handling in the MML web page, the event will

be sent to client.

If link to new document, the new document will be sent to client.

If link to new card or page within same document, the event with the card or page id will be

20 sent to client.

Result does not match:

If the implementation includes optional event handling in the MML web page, the nomatch

event with event handling information will be sent to client.

25 If the implementation does not include optional event handling in the MML web page, the

nomatch event with empty information will be sent to client.

3) Traditional Interaction:

URL request

30 New document will be sent to client

4) Exit Session:

If client exits, the exit message will be sent to server

As described above, various embodiments of the present invention provide a focus mechanism, synchronization mechanism and control mechanism, which are implemented by MML. MML extends the XHTML Basic by adding speech feature processing. Figure 8 shows the details of the MML element blocks used in one embodiment of the invention. When a content document is received by the Multi-modal server, a part of the MML element blocks will be sent to the client. The set of MML element blocks sent to the client are shown in Figure 8 within the dotted line 810. The whole document will be kept in the Multi-modal server.

The following is the detailed explanation for each MML element.

Elements	Attributes	Minimal Content Model
Html		(head, (body | (mml:card+)))
mml:card	id(ID), title(CDATA), style(CDATA)	(mml:onevent *,(Heading | Block | List)*, mml:speech?)
mml:speech	id(ID)	(mml:prompt *, mml:recog ?)
mml:recog	id(ID)	(mml:group ?, mml:sform*, mml:onevent*)
mml:group	id(ID), mode(speech | dtmf), accuracy(CDATA)	(mml:grammar, mml:link+, mml:onevent*)
mml:link	id(ID), value (CDATA), bind(IDREF)	EMPTY
mml:sform	id(ID), mode(speech | dtmf), accuracy(CDATA), bind(IDREF)	(mml:grammar, mml:input+, mml:onevent*)
mml:input	id(ID), value(CDATA), bind(IDREF)	EMPTY

mml:grammar	id(ID), src(CDATA)	PCDATA
mml:prompt	id(ID), type(text |tts | recorded), src(CDATA), loop(once | loop), interval(CDATA)	(PCDATA | mml:getvalue)*
mml:onevent	id(ID), type(match | nomatch | onload | unload |), phase(default |. capture), propagate(continue | stop), defaultaction(perform | cancel)	(mml:do)*
mml:do	id(ID), target(IDREF), href(CDATA), action(activate | reset)	EMPTY
mml:getvalue	id(ID), from(IDREF), at(client | server)	EMPTY

Referring still to Figure 8, each MML element in one embodiment is described in more detail below. By way of example, MML elements use the namespace identified by the "mml:" prefix.

The <card> element:

Elements	Attributes	Minimal Content Model
mml:card	id(ID), title(CDATA), style(CDATA)	(mml:onevent *,(Heading | Block | List)*, mml:speech?)

The function is used to divide the whole document into some cards or pages (segments). The client device will display one card at a time. This is optimized for small display devices and wireless transmission. Multiple card elements may appear in a single document. Each card element represents an individual presentation or interaction with the user.

The <mml:card> element is the one and only one element of MML that has relation to the content presentation and document structure.

The optional *id* attribute specifies the unique identifier of the element in the scope of the whole document.

The optional *title* attribute specifies the string that would be displayed on the title bar of the user agent when the associated card is loaded and displayed.

The optional *style* attribute specifies the XHTML inline style. The effect scope of the style is the whole card. But this may be overridden by some child XHTML elements, which could define their own inline style.

The <speech> element:

Elements	Attributes	Minimal Content Model
mml:speech	id(ID)	(mml:prompt *, mml:recog ?)

The <mml:speech> element is the container of all speech relevant elements. The child elements of <mml:speech> can be <mml:recog> and/or <mml:prompt>.

The optional *id* attribute specifies the unique identifier of the element in the scope of the whole document.

The <recog> element:

Elements	Attributes	Minimal Content Model
mml:recog	id(ID)	(mml:group ?, mml:sform*, mml:onevent*)

The <mml:recog> is the container of speech recognition elements.

The optional *id* attribute specifies the unique identifier of the element in the scope of the whole document.

5   The <group> element:

Elements	Attributes	Minimal Content Model
mml:group	id(ID), mode(speech | dtmf), accuracy(CDATA)	(mml:grammar, mml:link+, mml:onevent*)

The optional *id* attribute specifies the unique identifier of the element in the scope of the whole document.

10   The optional *mode* attribute specifies the speech recognition modes. Two modes are supported:

1. "speech" (default value)

"speech" mode is the default speech recognition mode.

2. "dtmf"

15   "dtmf" mode is used to receive telephony dtml signal. (This mode is to support traditional phones).

The optional **accuracy** attribute specifies the lowest accuracy of speech recognition that the page developers will accept. Following styles are supported:

20   1. "accept" (default value)

Speech recognizer sets whether the recognition output score is acceptable.

2. "xx" (eg. "60")

If the output score received from recognizer is equal to or greater than "xx"%, the recognition result will be considered as "match" and the "match" event

25   will be triggered. Otherwise, the result will be considered as "nomatch". The "nomatch" event will be triggered.

The <link> element:

Elements	Attributes	Minimal Content Model
mml:link	id(ID), value (CDATA), bind(IDREF)	EMPTY

The optional *id* attribute specifies the unique identifier of the element in the scope of the whole document.

The required value attribute specifies the <mml:link> element is corresponding to

5   which part of the grammar.

The required bind attribute specifies which XHTML hyperlink (such as <a>) is to be bound with.

The <sform> element:

10

Elements	Attributes	Minimal Content Model
mml:sform	id(ID), mode(speech | dtmf), accuracy(CDATA), bind(IDREF)	(mml:grammar, mml:input+, mml:onevent*)

The <mml:sform> element functions as the speech input form. It should be bound with the XHTML <form> element.

The optional *id* attribute specifies the unique identifier of the element in the scope of

15   the whole document.

The optional mode attribute specifies the speech recognition modes. Two modes are supported:
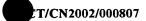
1. "speech" (default value)

"speech" mode is the default speech recognition mode.

20   2. "dtmf"

"dtmf" mode is used to receive telephony dtml signal. (This mode is to support traditional phones).

The optional *accuracy* attribute specifies the lowest accuracy of speech recognition

25   that the page developers will accept. Following styles are supported:

3. "accept" (default value)

      Speech recognizer sets whether the recognition output score is acceptable.

4. "xx" (eg. "60")

      If the output score received from recognizer is equal to or greater than "xx"%, the recognition result will be considered as "match" and the "match" event will be triggered. Otherwise, the result will be considered as "nomatch". The "nomatch" event will be triggered.

The <input> element:

Elements	Attributes	Minimal Content Model
mml:input	id(ID), value(CDATA), bind(IDREF)	EMPTY

      The <mml:input> element functions as the speech input data placeholder. It should be bound with XHTML <input>.

      The optional id attribute specifies the unique identifier of the element in the scope of the whole document.

      The optional *value* attribute specifies which part of the speech recognition result should be assigned to the bound XHTML <input> tag. If this attribute is not set, the whole speech recognition result will be assigned to the bound XHTML <input> tag.

The required bind attribute specifies which XHTML <input> in the <form> is to be bound with.

The <grammar> element:

Elements	Attributes	Minimal Content Model
mml:grammar	id(ID), src(CDATA)	PCDATA

      The <mml:grammar> specifies the grammar for speech recognition.

      The optional *id* attribute specifies the unique identifier of the element in the scope of the whole document.

The optional *src* attribute specifies the URL of the grammar document.

If this attribute is not set, the grammar content should be in the content of <mml:grammar>.

The <prompt> element:

Elements	Attributes	Minimal Content Model
mml:prompt	id(ID), type(text |tts | recorded), src(CDATA), loop(once | loop), interval(CDATA)	(PCDATA | mml:getvalue)*

The <mml:prompt> specifies the prompt message.

The optional *id* attribute specifies the unique identifier of the element in the scope of the whole document.

The optional *type* attribute specifies the prompt type. Three types are supported now in one embodiment:

1. "tts" (default value)

"tts" specifies the speech output is synthesized speech.

2. "recorded"

"recorded" specifies the speech output is prerecorded audio.

3. "text"

"text" specifies that the user agent should output the content in

Message Box.

If this attribute is set to "text", the client side user agent should ignore the "loop" and "interval" attribute.

If client side user agent has no TTS engine, it may override this "type" attribute from "tts" to "text".

The optional *src* attribute specifies the URL of the prompt output document.

If this attribute is not set, the prompt content should be in the content of <mml:promt>.

The optional *loop* attribute specifies how many times should the speech output be activated. Two modes are supported in one embodiment:

1. "once" (default value)

"once" means no loop.

2. "loop"

"loop" means the speech output will be played round and round, until the valid scope is

changed.

The optional *interval* attribute specifies the spacing time between two rounds of the speech output. It needs to be set only when the loop attribute is set to "loop".

Format:

"xxx" (eg. "5000")

User agent will wait "xxx" milliseconds between the two rounds of the speech output.

The <onevent> element:

Elements	Attributes	Minimal Content Model
mml:onevent	id(ID), type(match | nomatch | onload | unload), trigger(client | server), phase(default | capture), propagate(continue | stop), defaultaction(perform | cancel)	(mml:do)*

The <mml:onevent> element is used to intercept certain events.

The user agent (both the client and server) MUST ignore any <mml:onevent> element specifying a type that does not correspond to a legal event for the immediately enclosing element. For example: the server must ignore a <mml:onevent type="onload"> in a <mml:sform> element.

The *type* attribute indicates the name of the event

The optional id attribute specifies the unique identifier of the element in the scope of the whole document.

The required *type* attribute specifies the event type that would be handled. Following event types are supported in one embodiment:

5 1. "match"

"match" event occurs when the result of the speech recognition is accepted. This event type could only be valid when trigger attribute is set to "server".

2. "nomatch"

"nomatch" event occurs when the result of the speech recognition could not
10 be accepted. This event type could only be valid when trigger attribute is set to "server".

3. "onload"

"onload" occurs when certain display elements are loaded. This event type could only be valid when trigger attribute is set to "client".

15 4. "unload"

"unload" event occurs when certain display elements are unloaded. This event type could only be valid when trigger attribute is set to "client".

"Match" and "nomatch" event type can only be used with speech relevant elements.
20 "Onload" and "unload" event type can only be used with display elements.

The required *trigger* attribute specifies the event is desired to occur at client or server side.
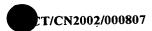
1. "client" (default value)
25 The event should occur at the client.

2. "server"

The event is desired to occur at server.

30 The optional *phase* attribute specifies when the <mml:onevent> will be activated by the desired event. If user agent (including client and server) supports MML Simple Content Events Conformance, this attribute should be ignored.

1. "default" (default value)

                <mml:onevent> should intercept the event during bubbling phase and on the target element.

2. "capture"

                <mml:onevent> should intercept the event during capture phase.

The optional *propagate* attribute specifies whether the intercepted event should continue propagating (XML Events Conformance). If the user agent (including client and server) supports MML Simple Content Events Conformance, this attribute should be ignored. The following modes are supported in one embodiment:

1. "continue" (default value)

The intercepted event will continue propagating.

2. "stop"

The intercepted event will stop propagating.

The optional *defaultaction* attribute specifies whether the default action for the event (if any) should be performed or not after handling this event by <mml:onevent>.

For instance:

The default action of a "match" event on an <mml:sform> is to submit the form. The default action of "nomatch" event on an <mml:sform> is to reset the corresponding <form> and give a "nomatch" message.

Following modes are supported in one embodiment:

1. "perform" (default value)

                The default action is performed (unless cancelled by other means, such as scripting, or by another <mml:onevent>).

2. "cancel"

The default action is cancelled.

The <do> element:

Elements	Attributes	Minimal Content Model
Mml:do	Id(ID),target(IDREF), href(CDATA), action(activate | reset)	EMPTY

The <mml:do> element is always a child element of an <mml:onevent> element. When the <mml:onevent> element intercepts a desired event, it will invoke the behavior specified by the contained <mml:do> element.

The optional *id* attribute specifies the unique identifier of the element in the scope of the whole document.

The optional *target* attribute specifies the id of the target element that will be invoked.

The optional *href* attribute specifies the URL or Script to the associated behavior. If the *target* attribute is set, this attribute will be ignored.

The optional *action* attribute specifies the action type that will be invoked on the target or URL.

1. "activate" (default value)

The target element or the URL will be activated. The final behavior is dependent on the target element type. For instance: If the target is a HYPERLINK, user agent will traverse it. If the target is a form, it will be submitted.

2. "reset"

The target element will be set to the initial state.

The <getvalue> element:

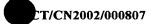| Elements	Attributes	Minimal Content Model
mml:getvalue	id(ID), from(IDREF), at(client|server)	EMPTY

The <mml:getvalue> element is a child element of <mml:prompt>. It is used to get the content from <form> or <sform> data placeholder.

The optional *id* attribute specifies the unique identifier of the element in the scope of the whole document.

The required *from* attribute specifies the identifier of the data placeholder.

The required *at* attribute specifies that the value to be assigned is at the client or the server:

1. "client" (default value)

The <mml:getvalue> get client side element value.

In this case, the *from* attribute should be set to a data placeholder of a <form>.

2. "server"

The <mml:getvalue> get server side element value.

5   In this case, the *from* attribute should be set to a data placeholder of a <sform>.

For example:

a.   <mml:getvalue at="client"> :


```
10    <mml:card>
          <mml:onevent type="onload">
                  <mml:do target="promptclient" type="activation">
          </mml:onevent >
           <form id="form01" action="other.mml" method="post">
15                    <input id="text1" type="text" name="ADD" value="ShangHai"/>
          </form>

          ...
          <mml:speech>
                  <mml:prompt id="promptclient" type="tts">
20                        the place you want to go, for example<mml:getvalue
from="text1" at="client" />
                  </mml:prompt>

                  ...
          </mml:speech>
25    </mml:card>
```


The process flow is as follows:

- The client user agent loads this card.

- The "onload" event will be triggered and then <mml:prompt> will be activated.

30 - Then <mml:getvalue> will be processed. The value of textbox "text1" will be retrieved by <mml:getvalue>. Here the initial value of the textbox is "ShangHai".

- Finally, the client will talk to the user: "The place you want to go, for example: ShangHai" .


```
35    b.  <mml:getvalue at="server">:
              <mml:card>
                  <form id="form01" action="other.mml" method="post">
                      <input id="text1" type="text" name="add" value="ShangHai"/>
```

```
            </form>


        <mml:speech>
            <mml:prompt id="promptServer" type="tts">
                The place you want to go<mml:getvalue from="stext1"
at="server" />
            </mml:prompt>
            <mml:recog>
                <mml:sform id="sform01" bind="form01">
                    <mml:grammar src="Add.gram"/>
                    <mml:input id="stext1" value="#add"
bind="text1"/>

                    <mml:onevent type="match">
                        <mml:do target="promptServer"
type="activation"/>

                    </mml:onevent>
                </mml:sform>
            </mml:recog>
    < /mml:speech>
        </mml:card>
```

The process flow is as follows:

- User inputs an utterance and the speech recognition will be performed.
- If the speech output score of the speech recognition is acceptable, the "match" event will be triggered.
- Before submitting the form, server will process the "match" event handler (<mml:onevent>) first. Then, an event message will be sent to the client as follows:

```
    <event type="match">
        <do target="promptServer">
        <input id="stext1" value="地点"/>
         <!—It's according to the recognition result -->
    </event>
```

- Then, when the client processes the <mml:prompt> and <mml:getvalue at="server">, the value received from the server will be assigned to <mml:getvalue> element.
- Finally, the client will talk to the user. The talk content is related to the speech recognition result.

The following section describes the flow of client and server interaction in the system for multi-modal web interaction over wireless network according to one embodiment of the present invention.

Unlike traditional web interaction and telephony interaction, the system of the
5 present invention supports multi-modal web interaction. Because the main speech recognition processing job is handled by the server, the multi-modal web page will be interpreted at both the client and the server side. The following is an example of the simple flow of client and server interaction using an embodiment of the present invention.

10 <User>: Select hyperlinks, submit a form (traditional web interaction) or press the "Talk Button" and input an utterance (speech interaction).

<Client>: In the case of traditional web interaction, the client transmits a request to the server for a new page or submits the form. In case of speech interaction, the client
15 determines which active display element is to be focused and the ID of the focused display element, captures speech, extracts speech features, and transmits the id of focused display element, the extracted speech features and other information such as URL of the current page to the server. Then, the client waits for a response.

20 <Server>: In the case of traditional web interaction, the server retrieves the new page from a cache or web server and sends it to the client. In the case of speech recognition, the server receives the id of the focused display element and builds the correct grammar. Then, speech recognition will be performed. According to the result of speech recognition, the server will do specific jobs and send events or new pages to the client. Then, the server waits for a new
25 request from the client.

<Client>: Client will load the new page or handle events.

Thus, an inventive multi-modal web interaction approach with focus mechanism,
30 synchronize mechanism and control mechanism implemented by MML is disclosed. The scope of protection of the claims set forth below is not intended to be limited to the particulars described in connection with the detailed description of various embodiments of the present invention provided herein.